

Árvores



Definição

Uma árvore enraizada, T , é um conjunto finito de elementos chamados vértices (ou nós), tal que:

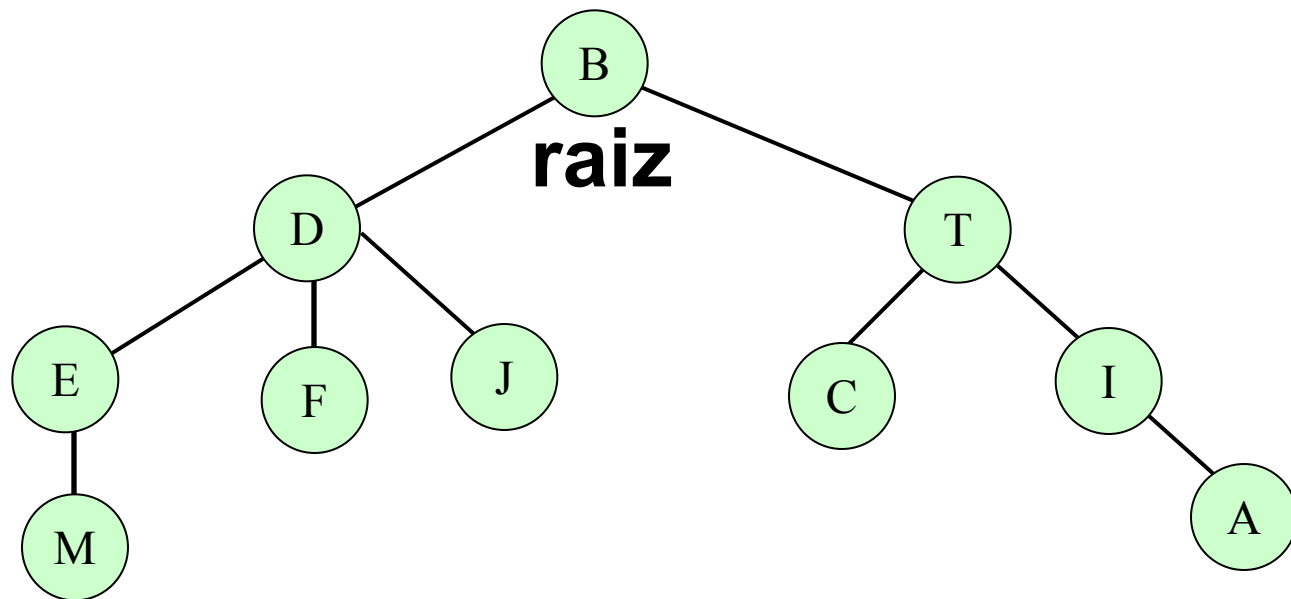
- i) $T = \emptyset$ (a árvore é dita vazia) ou
- ii) Existe um nó especial, r , dito *raiz* da árvore de T e os nós restantes constituem um único conjunto vazio ou são divididos em $m \geq 1$ conjuntos disjuntos não vazios, chamados de *subárvores* de r , cada qual uma árvore.



Cajueiro de Pirangi

Representação

A representação mais comum é a hierárquica.

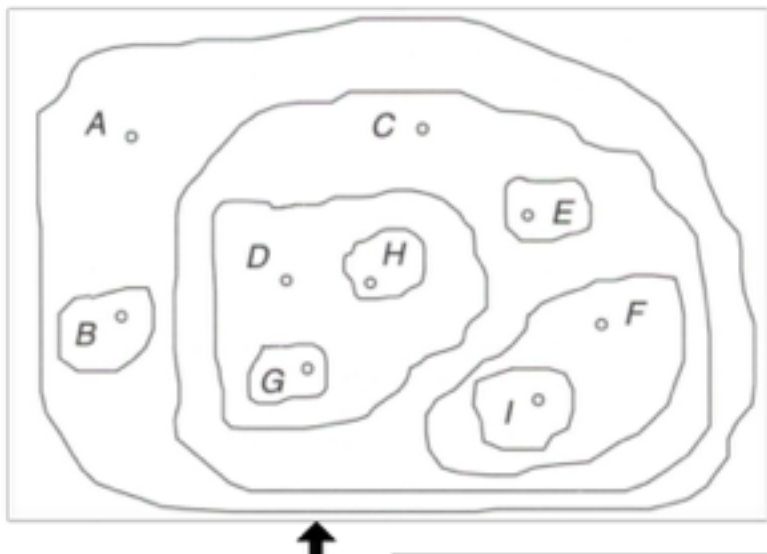


Representação

**Que outras representações
existem para árvores além da
hierárquica?**



Representação

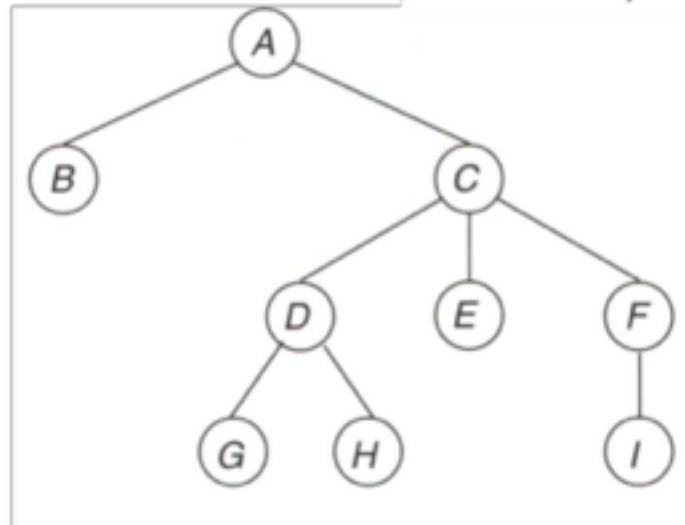


Inclusão ↗

A _____
B _____
C _____
D _____
G _____
H _____
E _____
F _____
I _____

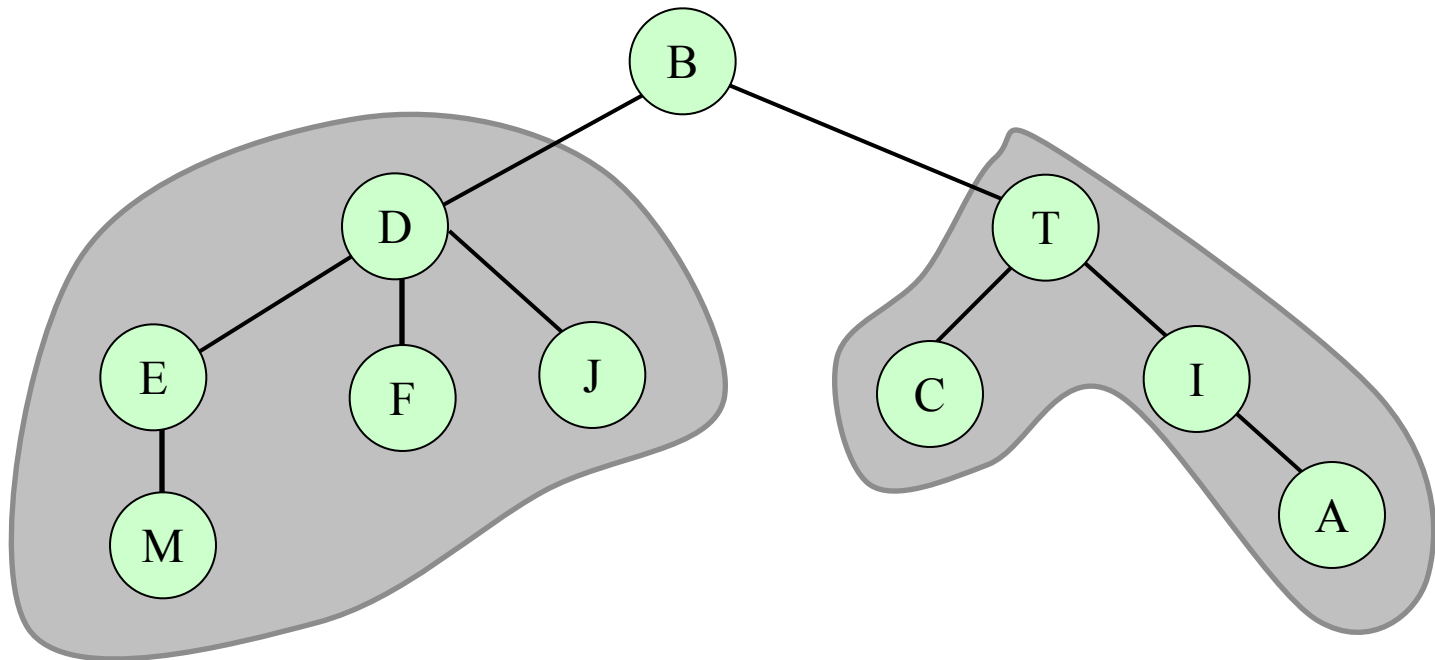
Barras ↗

Hierárquica ↘



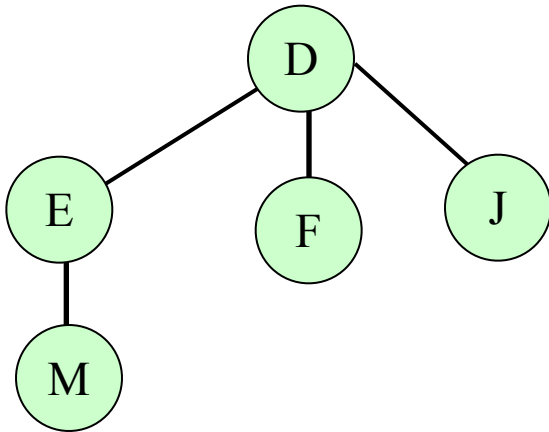
Definições

Subárvores da raiz

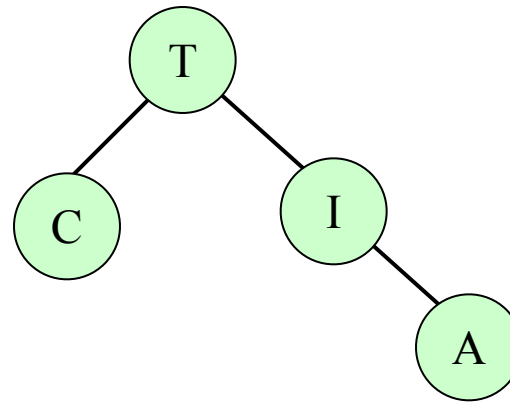


Definições

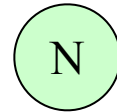
Floresta = um conjunto de árvores



Árvore 1



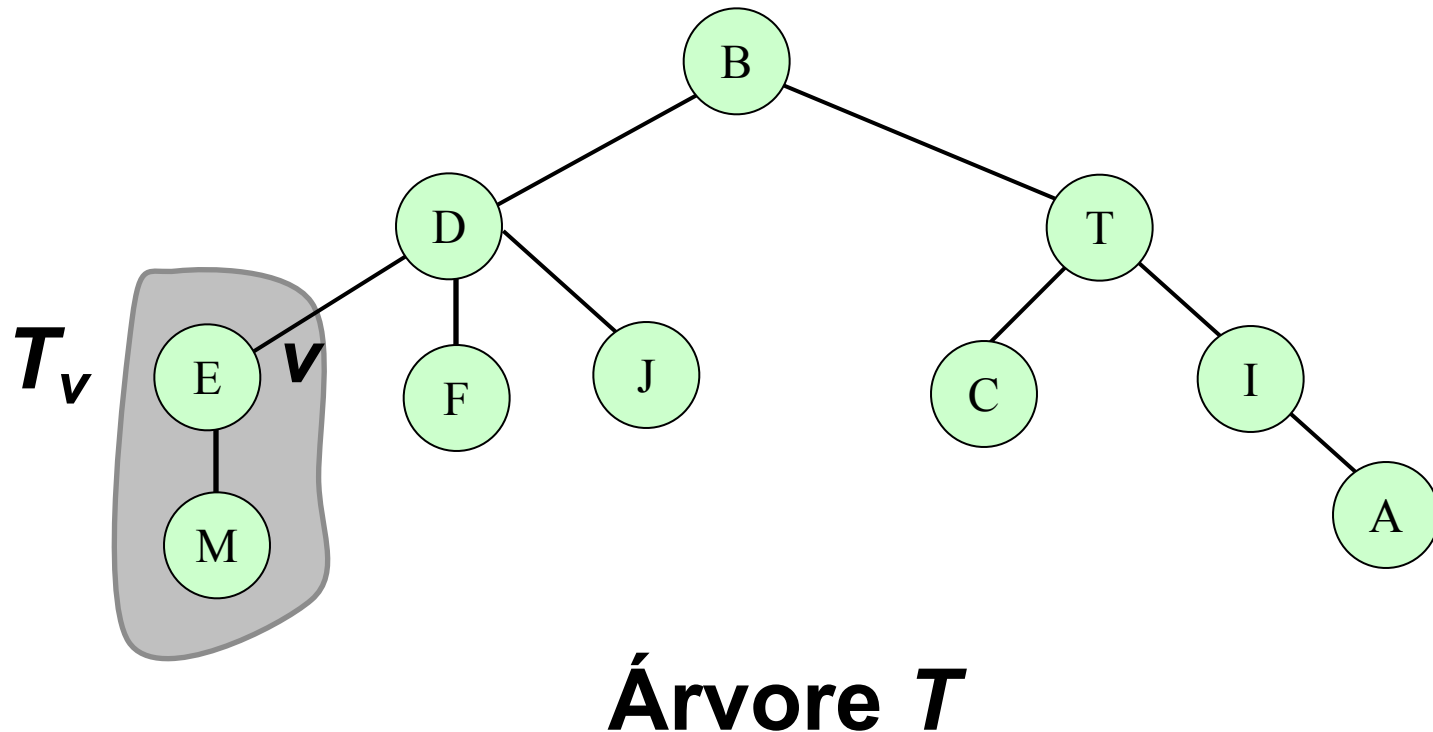
Árvore 2



Árvore 3

Definições

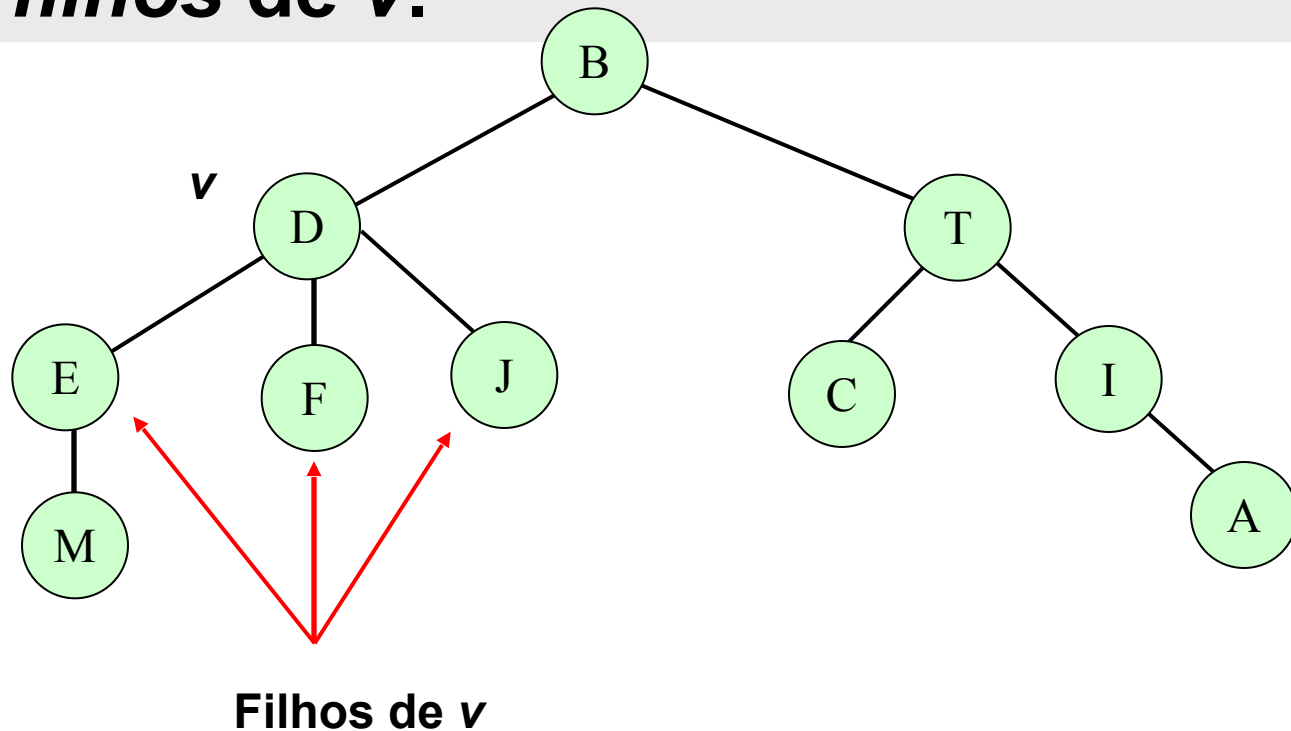
Seja v um nó de T , a notação T_v representa a subárvore de T com raiz v .



Definições

Seja v o nó raiz de T_v .

Os nós raízes w_1, \dots, w_j das subárvores de T_v são os *filhos* de v .

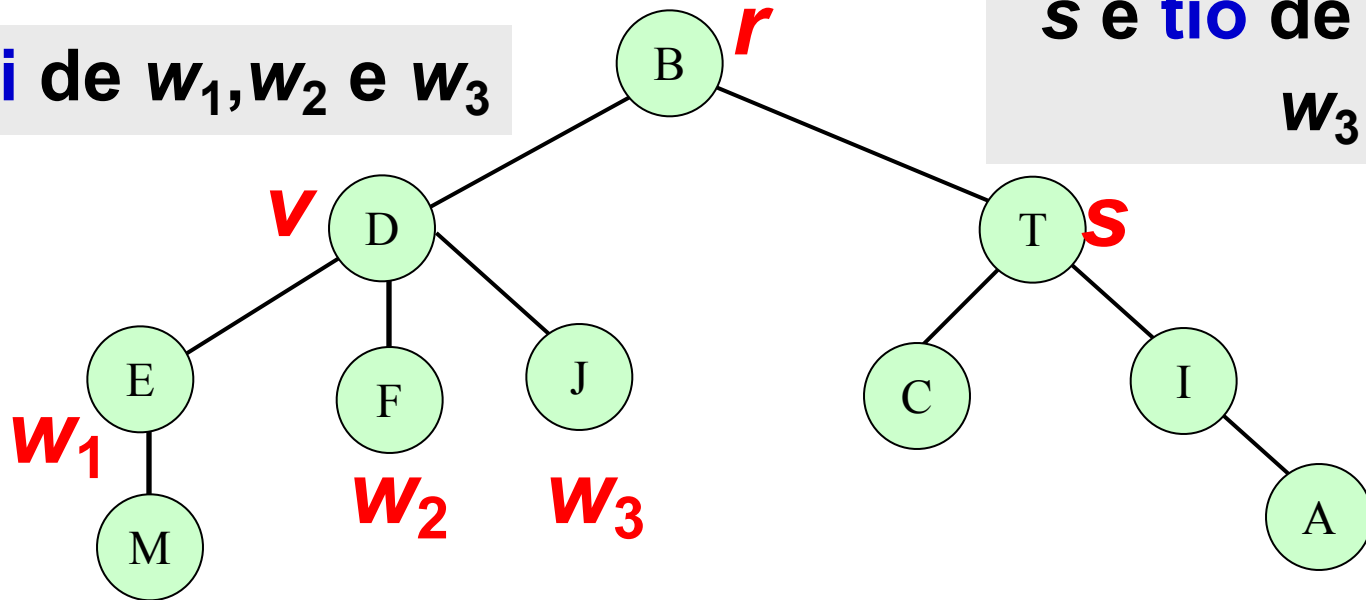


Definições

r é **avô** de w_1, w_2 e w_3

v é **pai** de w_1, w_2 e w_3

s é **tio** de w_1, w_2 e w_3

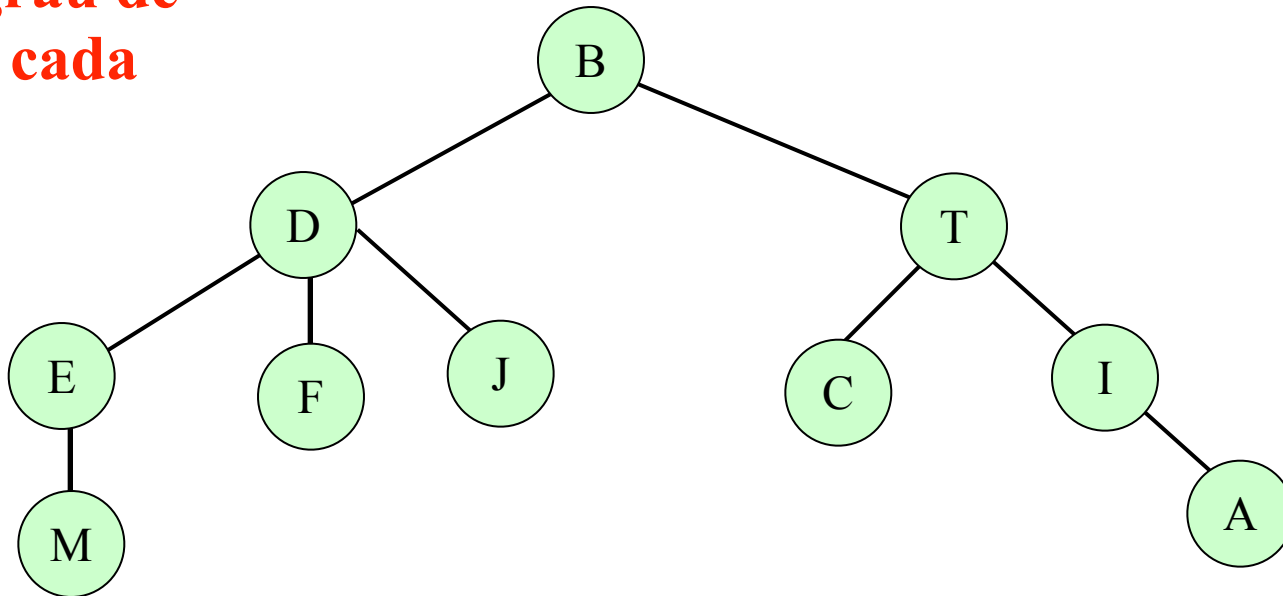


w_1, w_2 e w_3 são **irmãos**

Definições

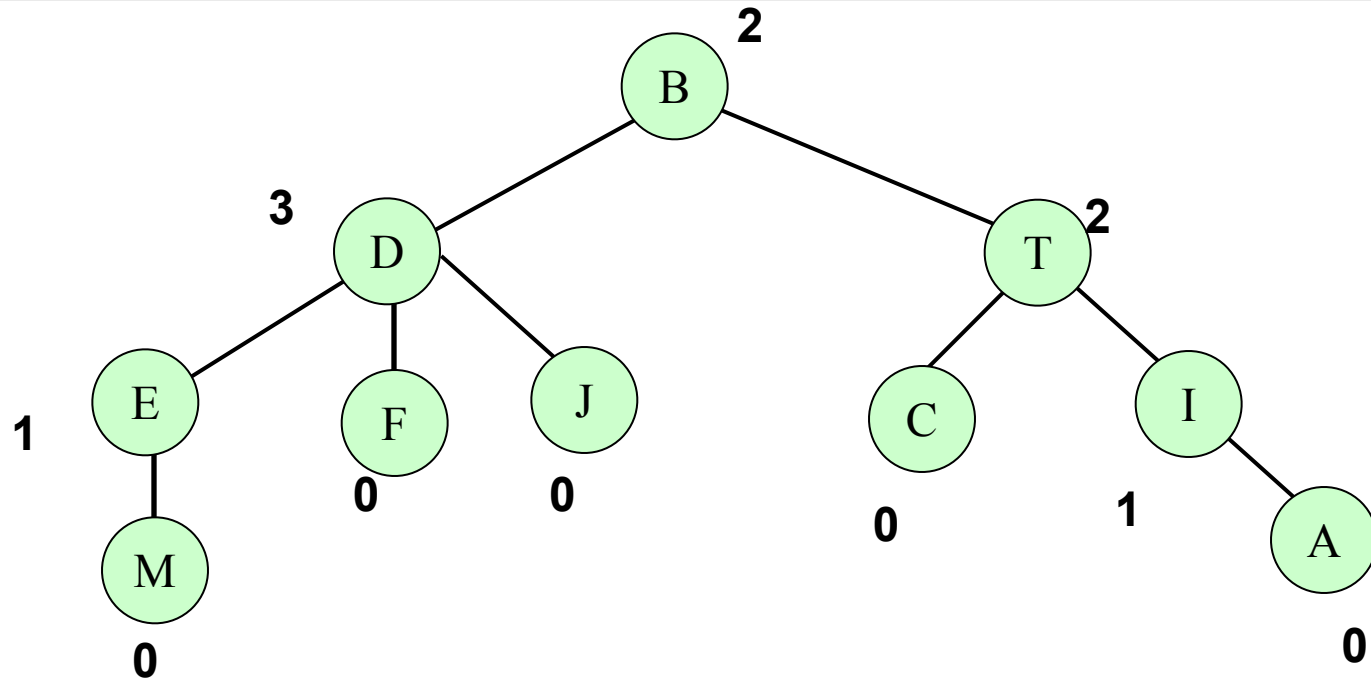
O número de filhos de um nó v é chamado **grau de saída de v**

Qual o grau de saída de cada nó desta árvore?



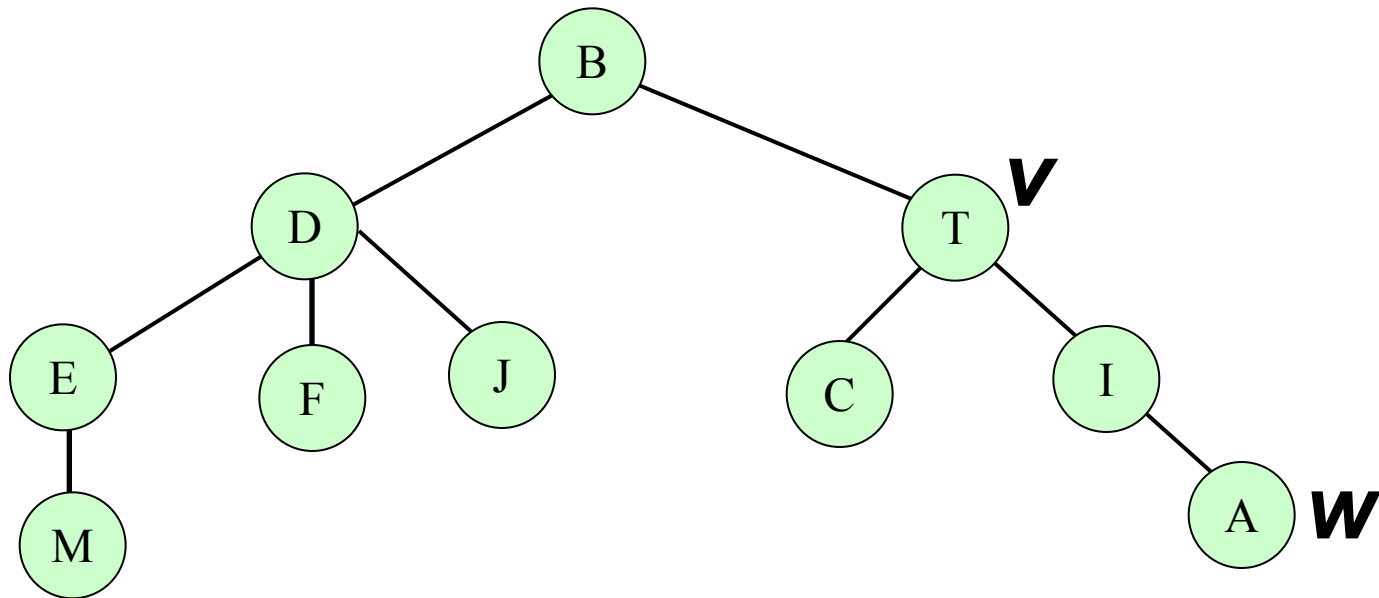
Definições

O número de filhos de um nó v é chamado **grau de saída de v**



Definições

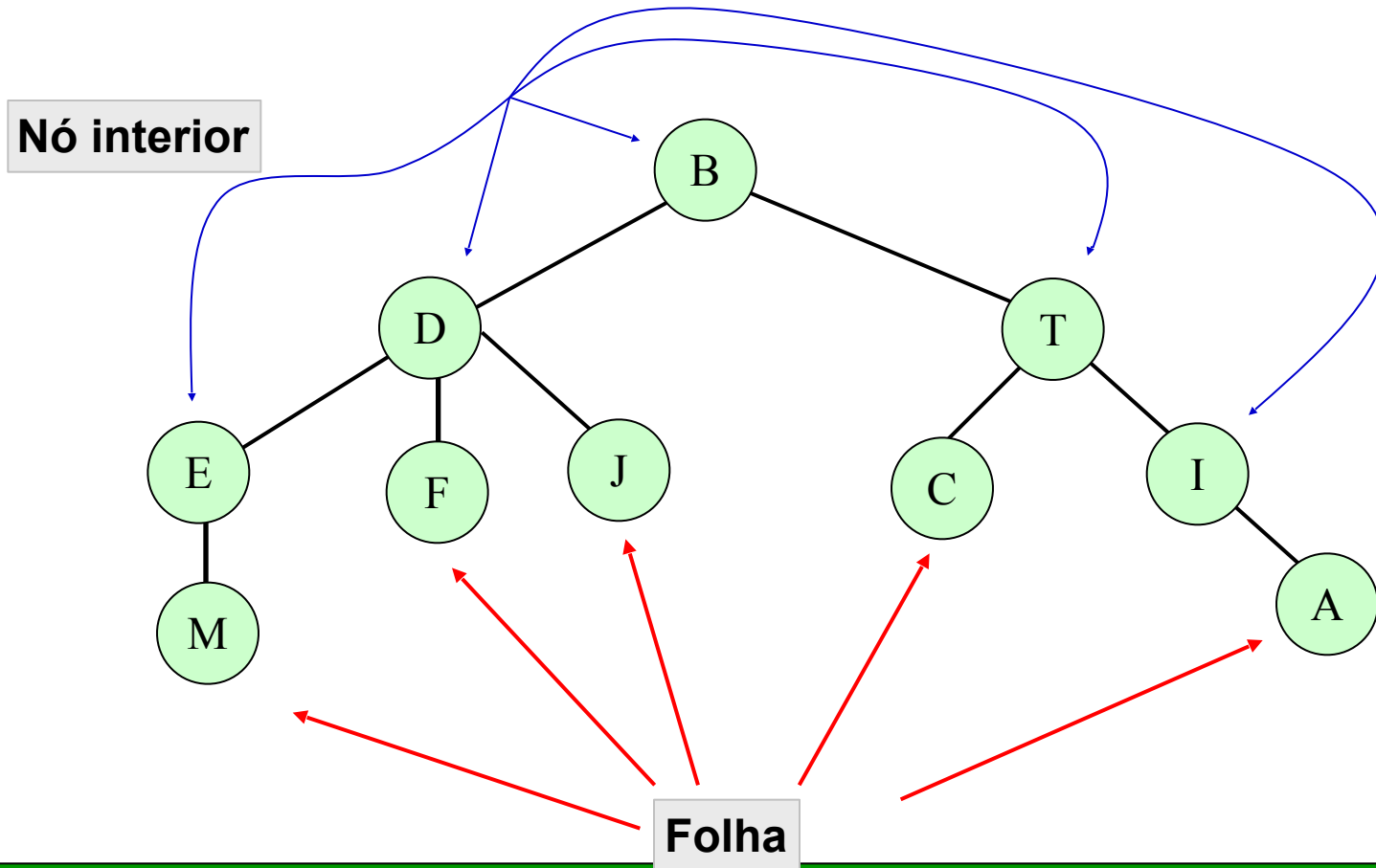
Se w pertence a T_v , então w é descendente de v e v é ancestral de w



Observação: se $v \neq w$, diz-se ancestral próprio ou descendente próprio

Definições

Um nó com **grau de saída igual a 0** é chamado **folha**



Teorema

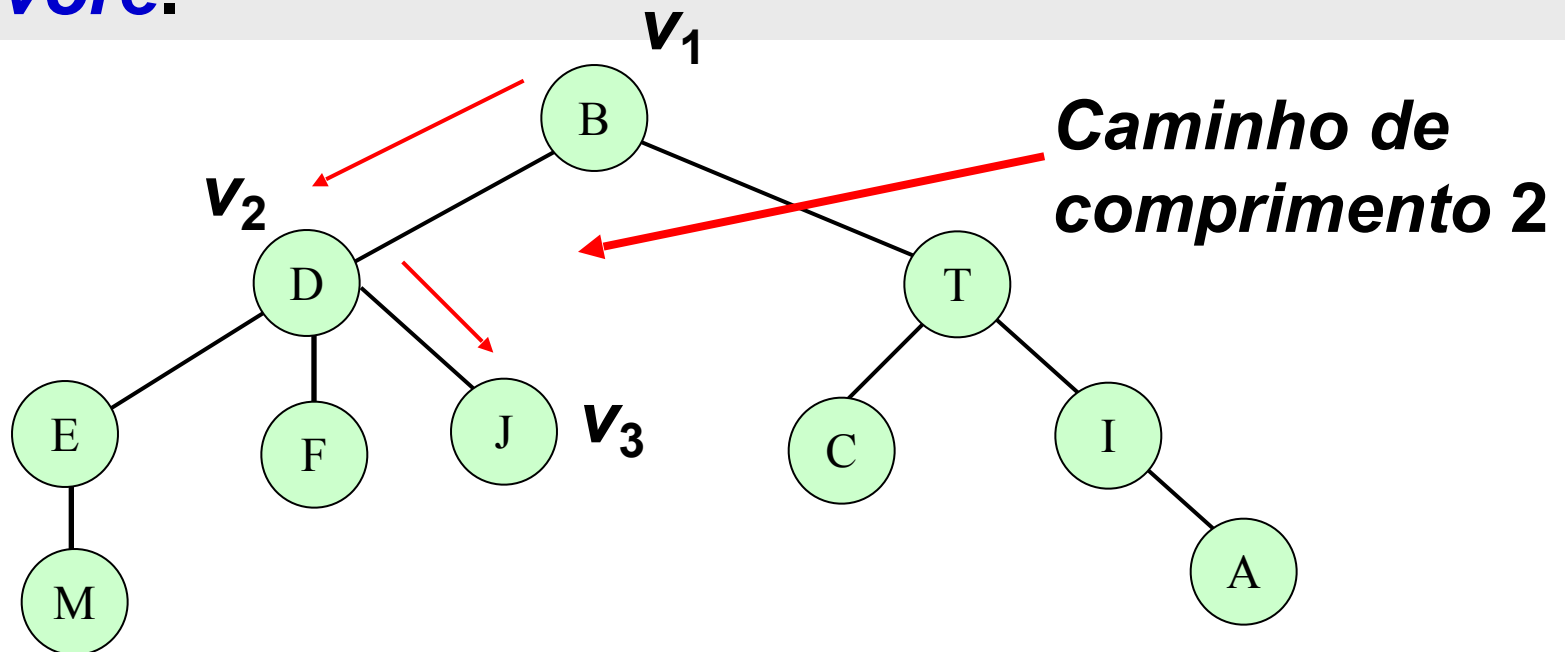
Toda árvore com número de nós maior que 1 ($n > 1$) possui no mínimo 1 e no máximo $n - 1$ folhas.

Exercício: Prove este teorema por indução



Definições

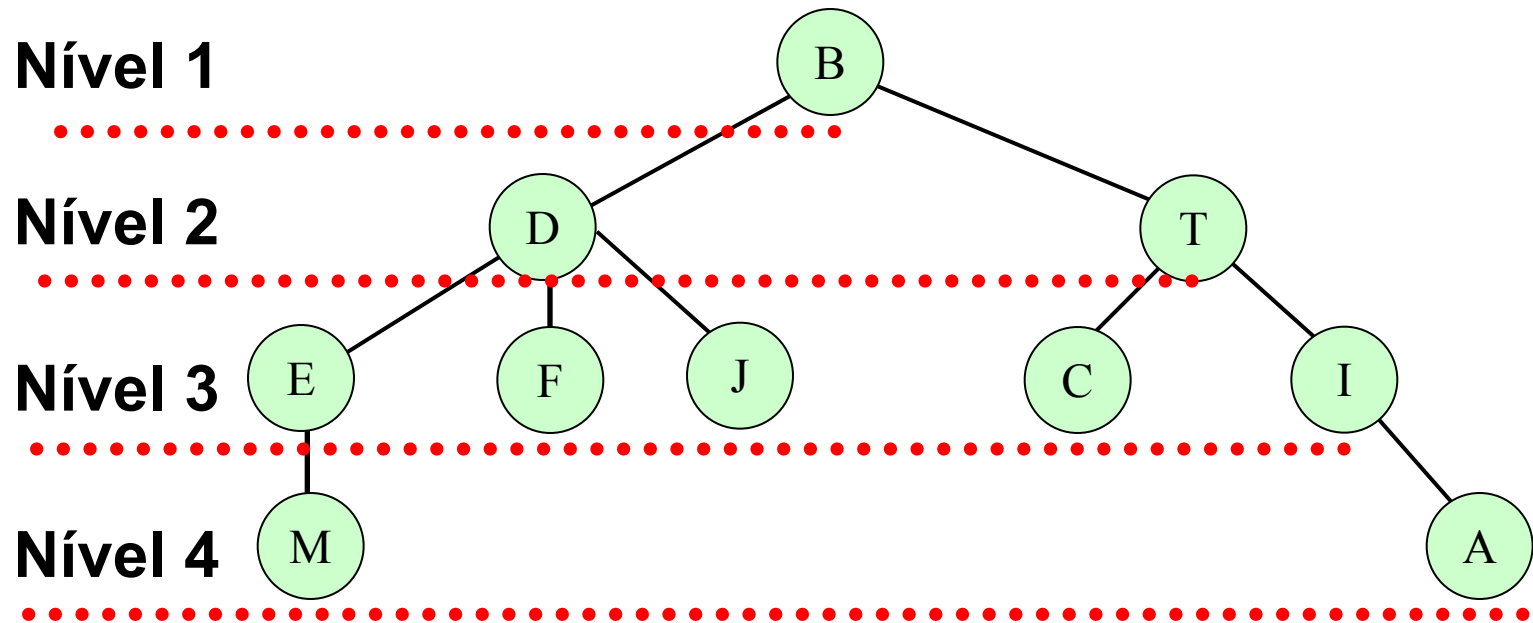
Uma sequência de nós distintos v_1, v_2, \dots, v_k tal que para dois nós consecutivos existe sempre a relação *é filho de* (*é pai de*) é dita *um caminho na árvore*.



Comprimento do caminho = $k-1$

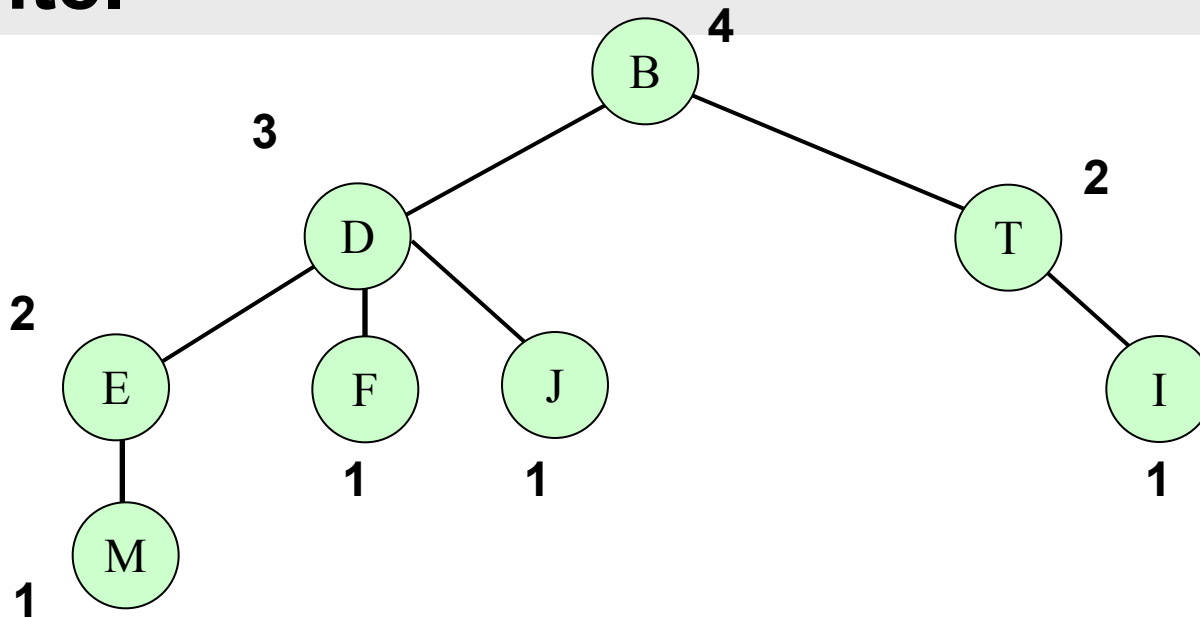
Definições

Nível do nó v é o número de nós no caminho entre a raiz da árvore e v .



Definições

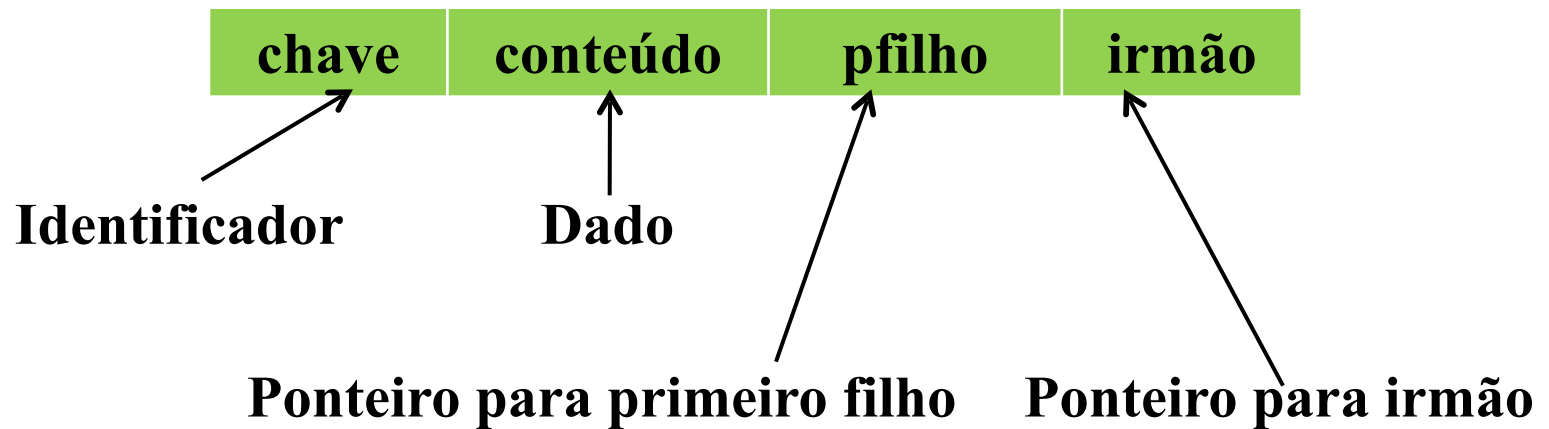
Altura do nó v é o número de nós no caminho entre v e seu descendente mais distante.



A altura da árvore, $h(T)$, é dada pela altura da raiz (ou pelo nível máximo de seus nós)

Árvore

Nó da árvore



Árvore

Nó da árvore

chave

conteúdo

pfilho

irmão

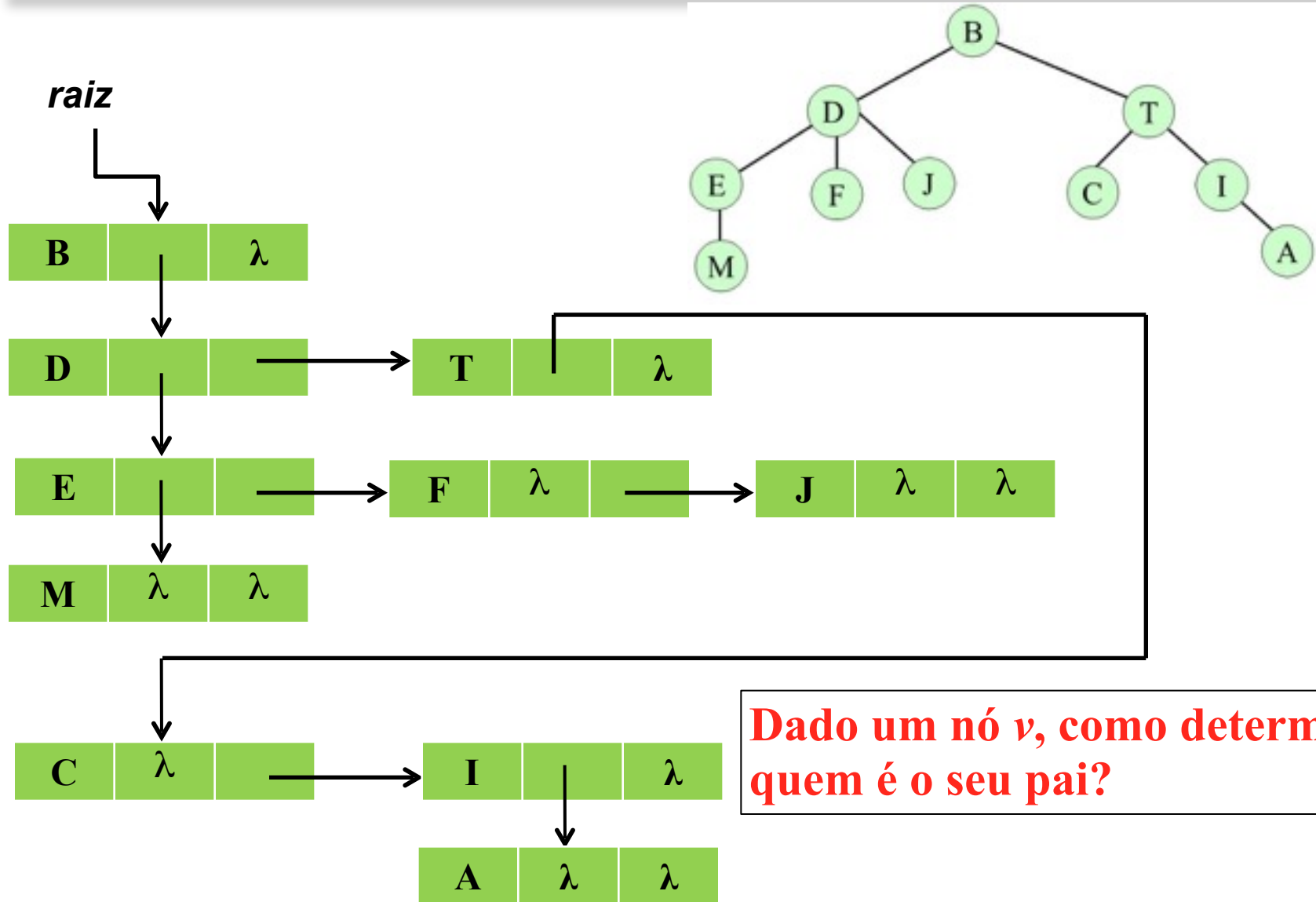
Exemplo C:

```
struct nodo {  
    int chave;  
    int conteudo;  
    struct nodo *pfilho;  
    struct nodo *irmao;  
};  
typedef struct nodo no;
```

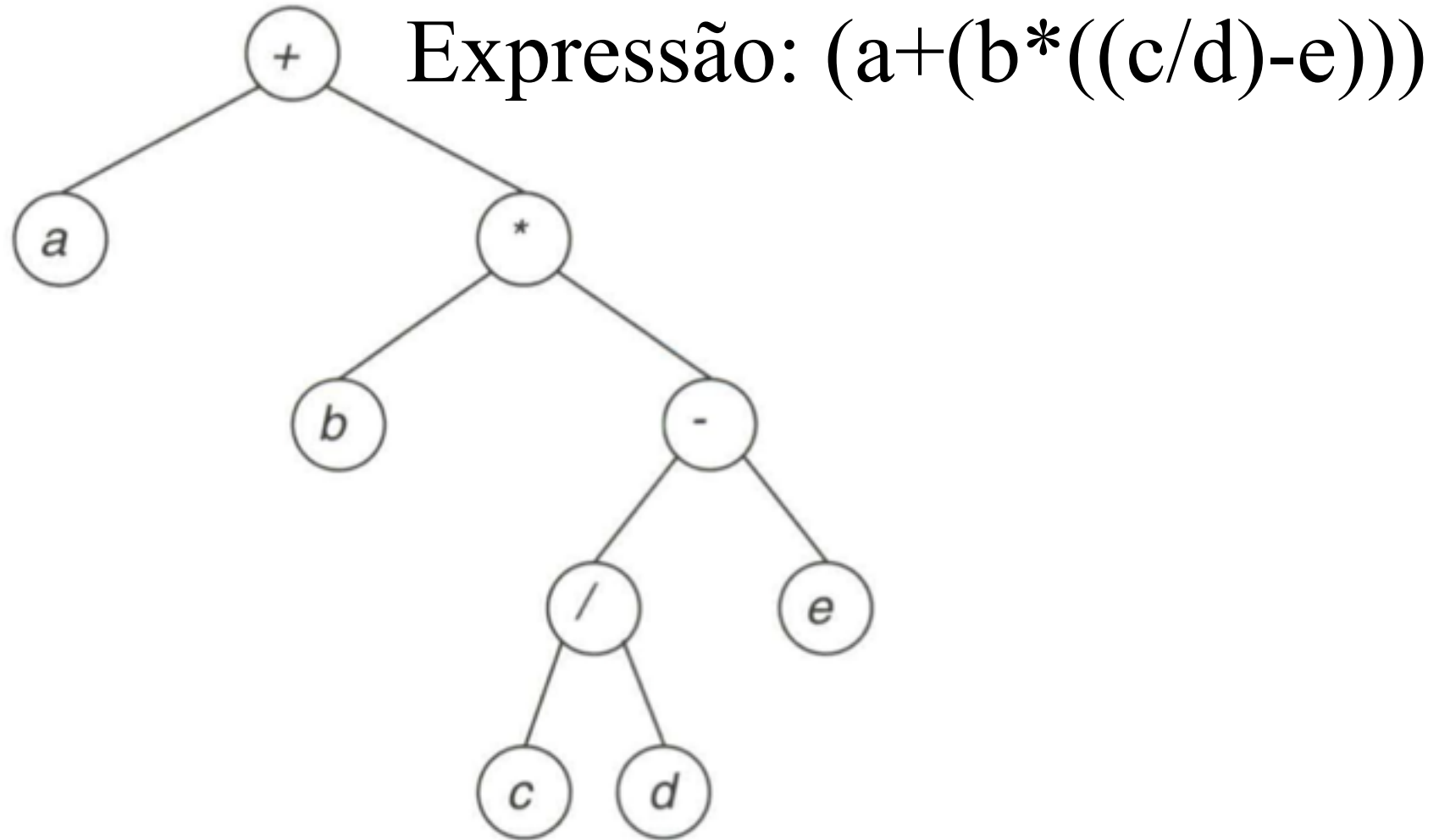
```
typedef no *pont_no;
```

Observação: O campo *chave* pode ser usado como chave e conteúdo

Árvore



Expressão aritmética

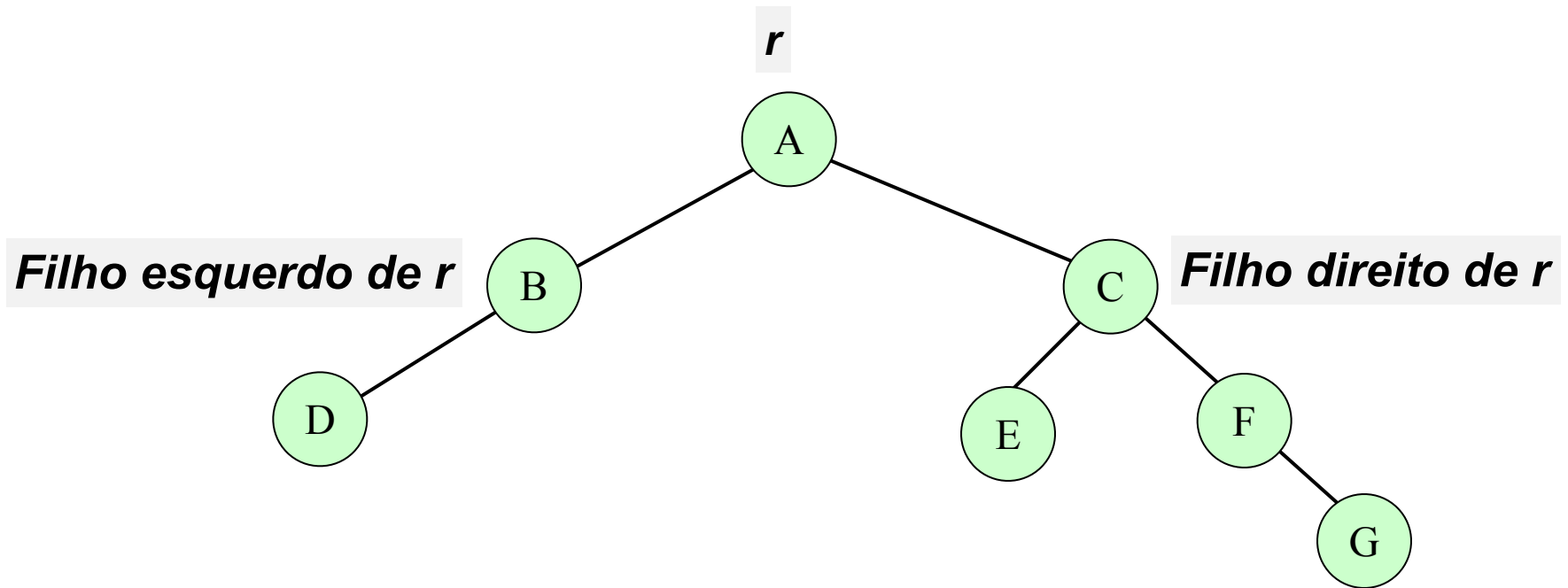


Árvore Binária

Uma árvore binária, T , é um conjunto finito de elementos chamados vértices (ou nós), tal que:

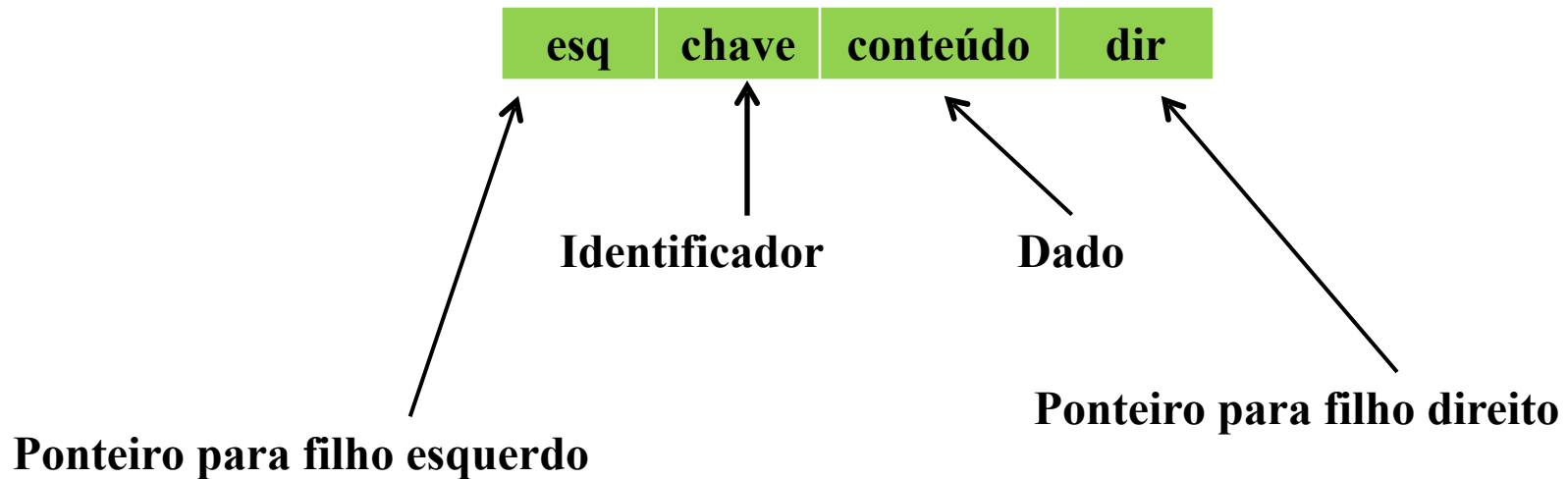
- i) $T = \emptyset$ (a árvore é dita vazia) ou
 - ii) Existe um nó especial, r , dito *raiz* da árvore T e os nós restantes são divididos em dois conjuntos disjuntos, a subárvore esquerda e direita de r , cada qual uma árvore binária.
-

Árvore Binária



Árvore Binária

Nó da árvore



Árvore Binária

Nó da árvore

esq

chave

conteudo

dir

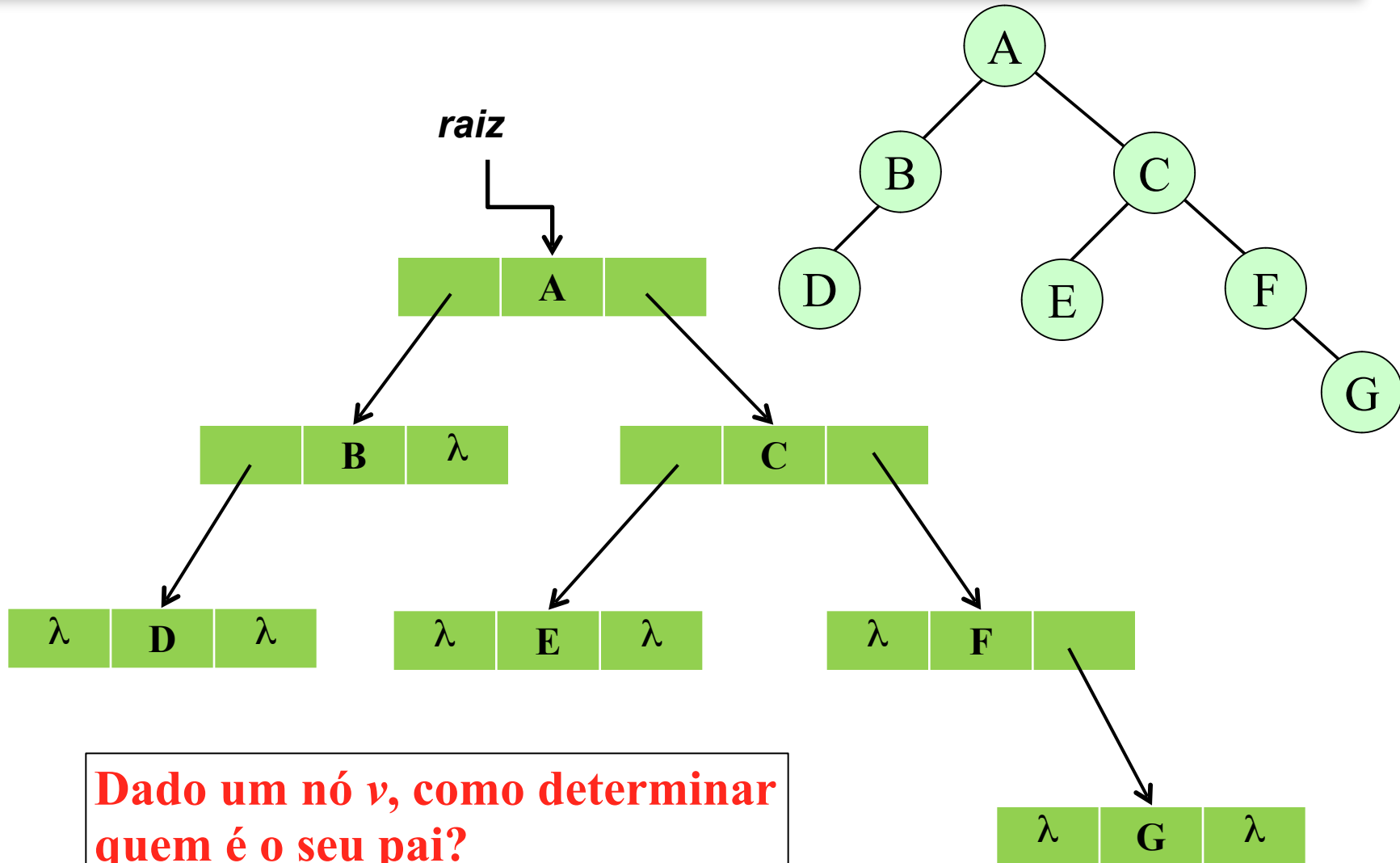
Exemplo C:

```
struct nodo {  
    int chave;  
    int conteudo;  
    struct nodo *esq;  
    struct nodo *dir;  
};  
typedef struct nodo no;
```

```
typedef no *pont_no;
```

Observação: O campo *chave* pode ser usado como chave e conteúdo

Árvore Binária



Árvore Binária

Lema 1

O número de subárvores vazias de uma árvore binária com n nós é
$$n + 1$$

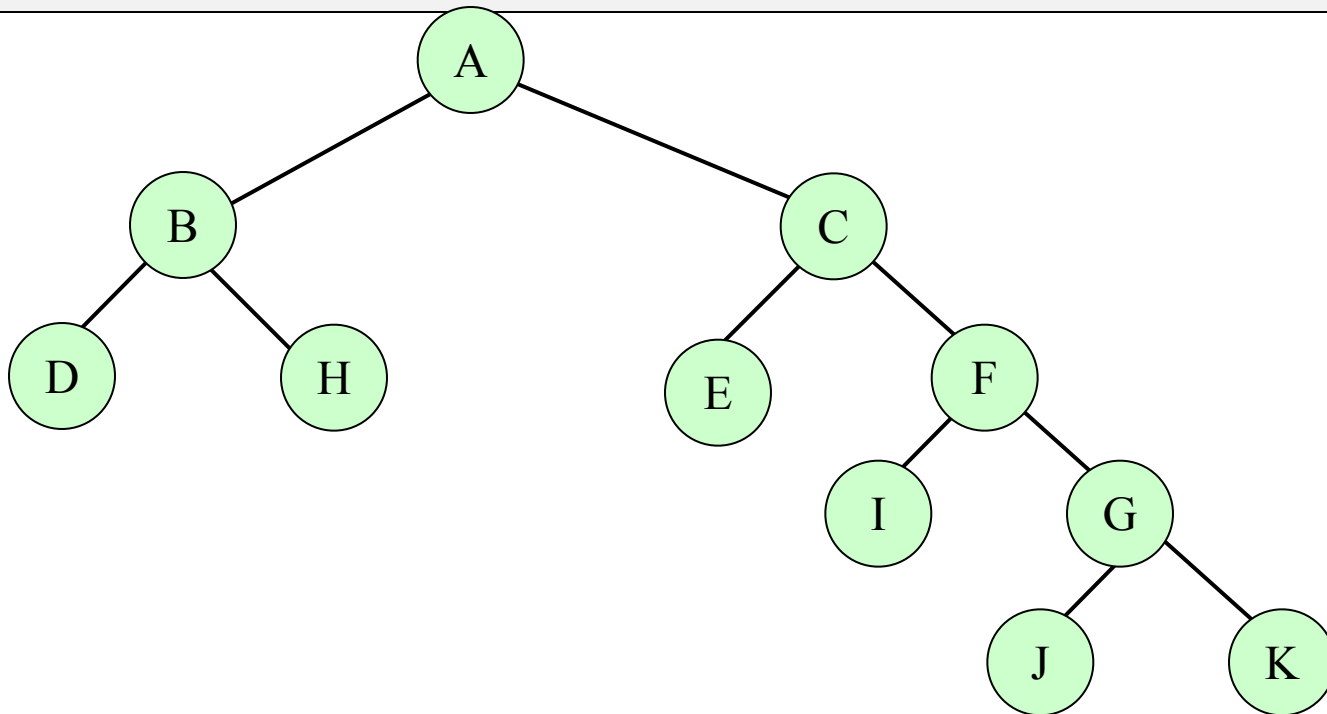
Exercício: Prove o Lema 1



Árvores Binárias Especiais

Árvore Estritamente Binária

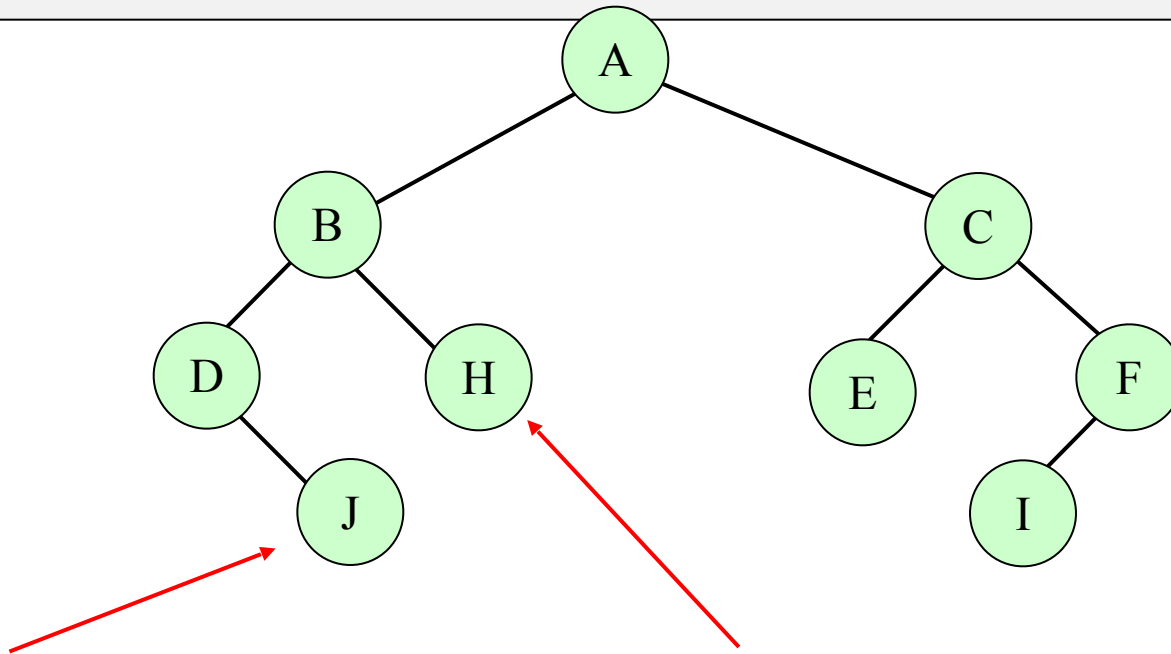
Todo nó possui 0 ou 2 filhos



Árvores Binárias Especiais

Árvore Binária Completa

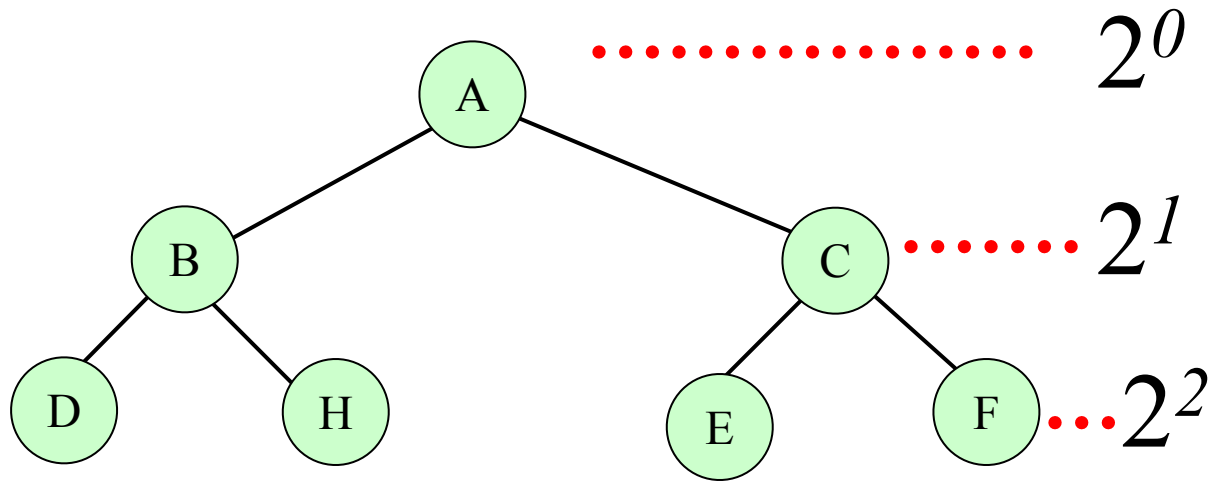
Se v é um nó com uma subárvore vazia, então v está no último ou no penúltimo nível de T .



Árvores Binárias Especiais

Árvore Binária Cheia

Se v é um nó com uma subárvore vazia, então v está no último nível de T .

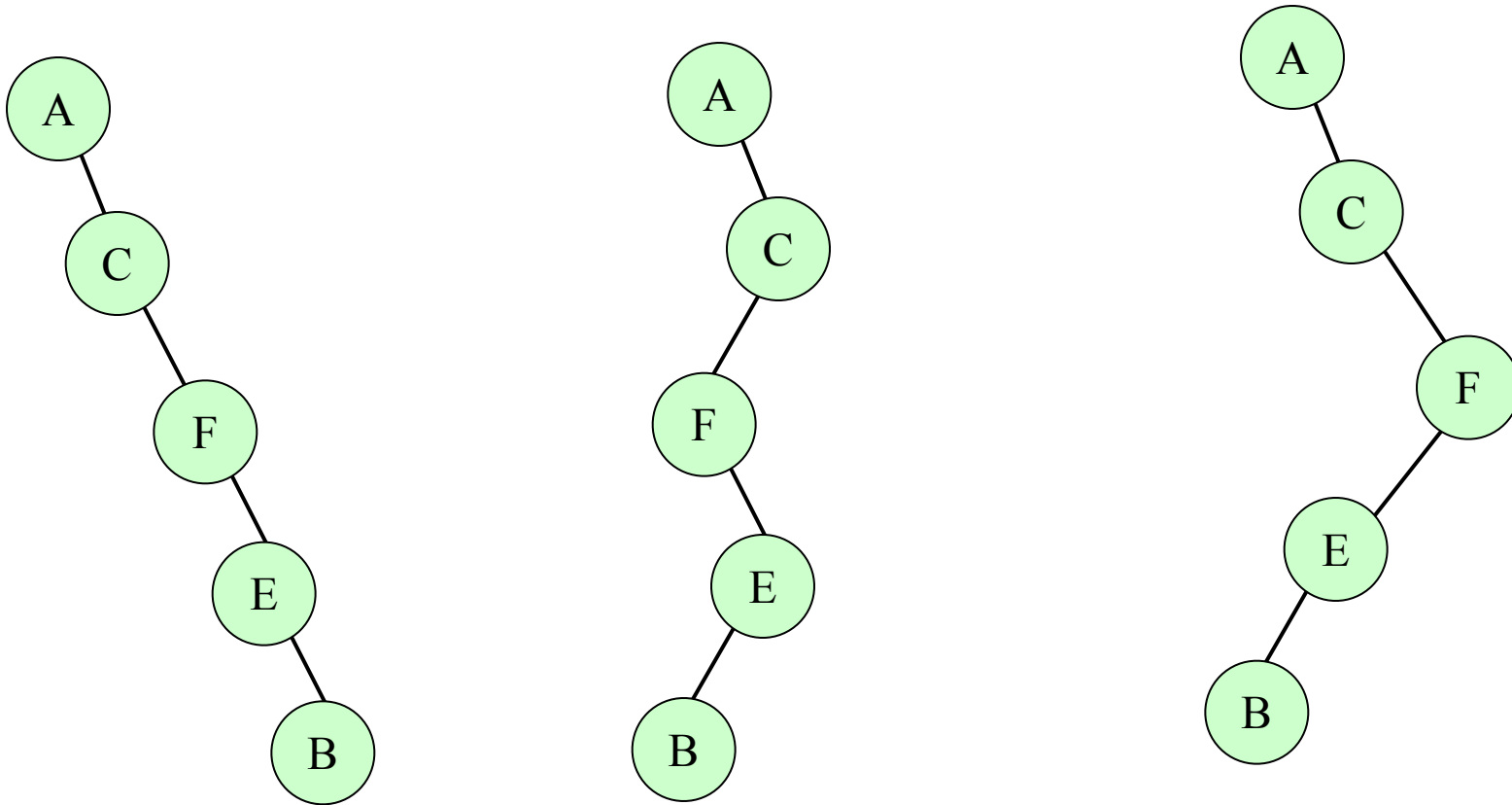


Total de nós = $2^h - 1$

Obs. Toda árvore cheia é estritamente binária e completa

Árvores Binárias Especiais

Árvores Ziguezague



Altura máxima para um número fixo de nós

Árvores Binárias Especiais

Lema 2

Considere T uma árvore binária completa com $n > 0$ nós.

Então T possui altura h mínima.

Além disso, $h = 1 + \lfloor \log n \rfloor$.

Exercício: Prove o lema 2



Árvores Binárias Especiais

Lema 3

Seja T uma árvore binária completa com n nós e altura h .

$$\text{Então, } 2^{h-1} \leq n \leq 2^h - 1$$

Exercício: Prove o lema 3



Percurso em Árvore Binária

Um **percurso** é uma visita sistemática a cada nó da árvore.

Corresponde a conhecer a informação contida no nó, percorrer a subárvore esquerda e percorrer a subárvore direita.



Percurso em Árvore Binária

Considerando da esquerda para a direita

Considerando que visitar(raiz) significa imprimir o conteúdo do nó.

Pré-ordem

visitar(raiz)

percorrer subárvore esquerda da raiz em pré-ordem

percorrer subárvore direita da raiz em pré-ordem

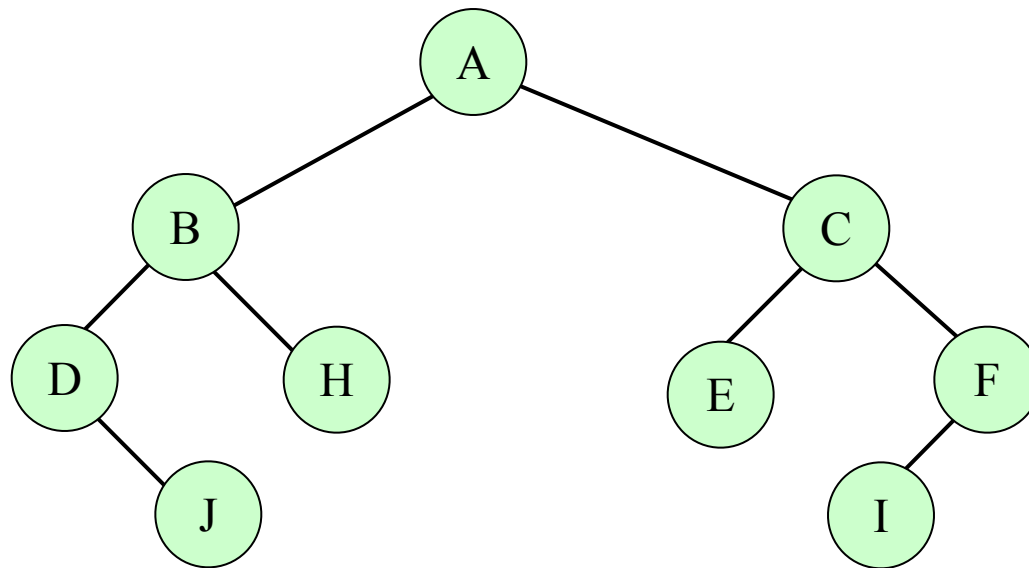
Percurso em Árvore Binária

Exemplo: Pré-ordem

visitar(raiz)

percorrer subárvore esquerda da raiz em pré-ordem

percorrer subárvore direita da raiz em pré-ordem



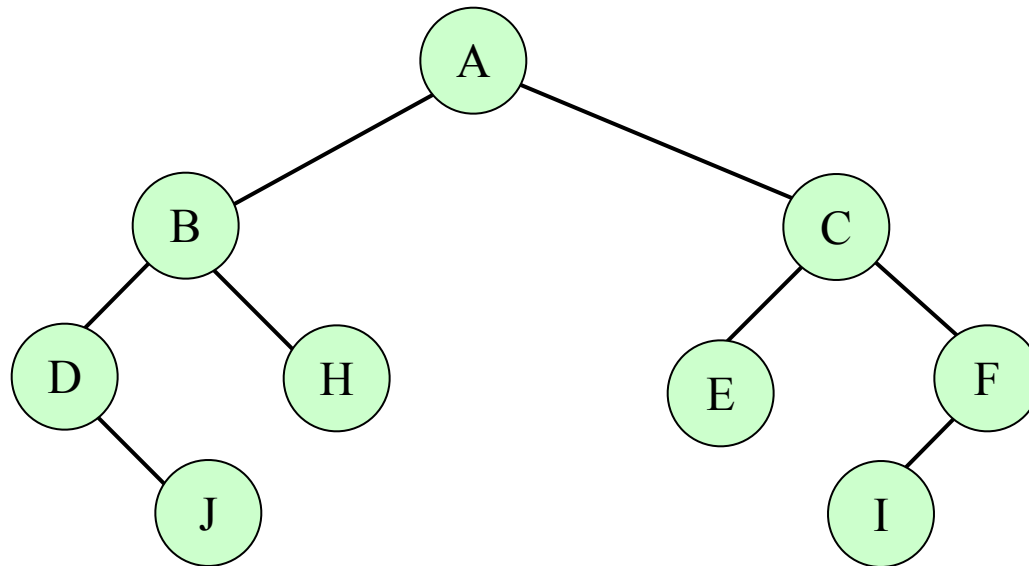
Percurso em Árvore Binária

Exemplo: Pré-ordem

visitar(raiz)

percorrer subárvore esquerda da raiz em pré-ordem

percorrer subárvore direita da raiz em pré-ordem



Pré-ordem: A B D J H C E F I

Percurso em Árvore Binária

Principal

```
se  $pt \neq \lambda$   
  pre_ordem( $pt$ )
```

Algoritmo pre_ordem(pont_no pt)

```
visitar( $pt$ )  
se  $pt \uparrow .esq \neq \lambda$   
  pre_ordem( $pt \uparrow .esq$ )  
se  $pt \uparrow .dir \neq \lambda$   
  pre_ordem( $pt \uparrow .dir$ )
```

Percurso em Árvore Binária



Exercício

Qual a complexidade de `pre_ordem`, considerando que o procedimento visita é constante?



Percurso em Árvore Binária



Exercício

Qual a complexidade de `pre_ordem`, considerando que o procedimento visita é constante?

$O(n)$



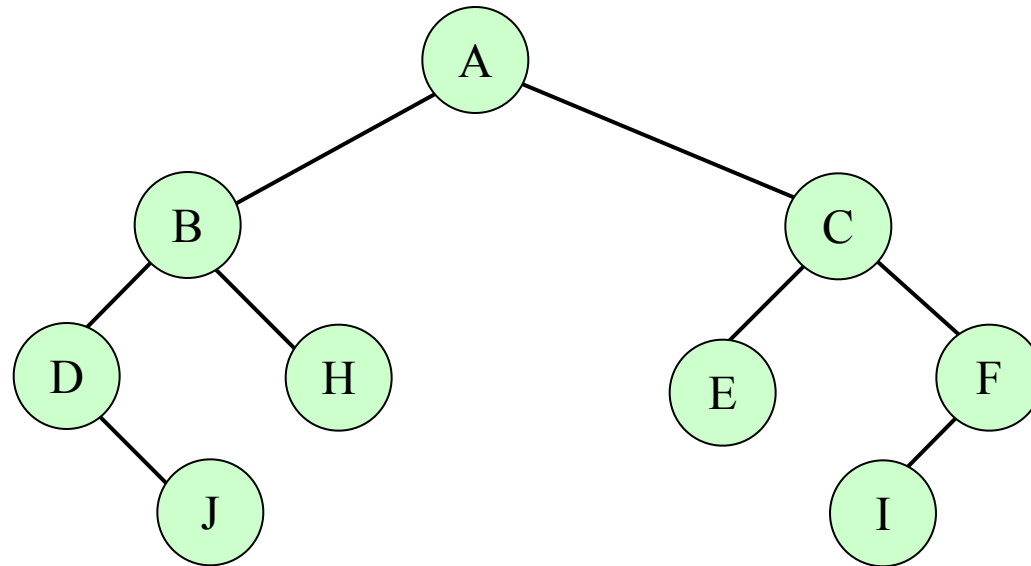
Percurso em Árvore Binária

Ordem Simétrica

percorrer subárvore esquerda da raiz em ordem simétrica

visitar(raiz)

percorrer subárvore direita da raiz em ordem simétrica



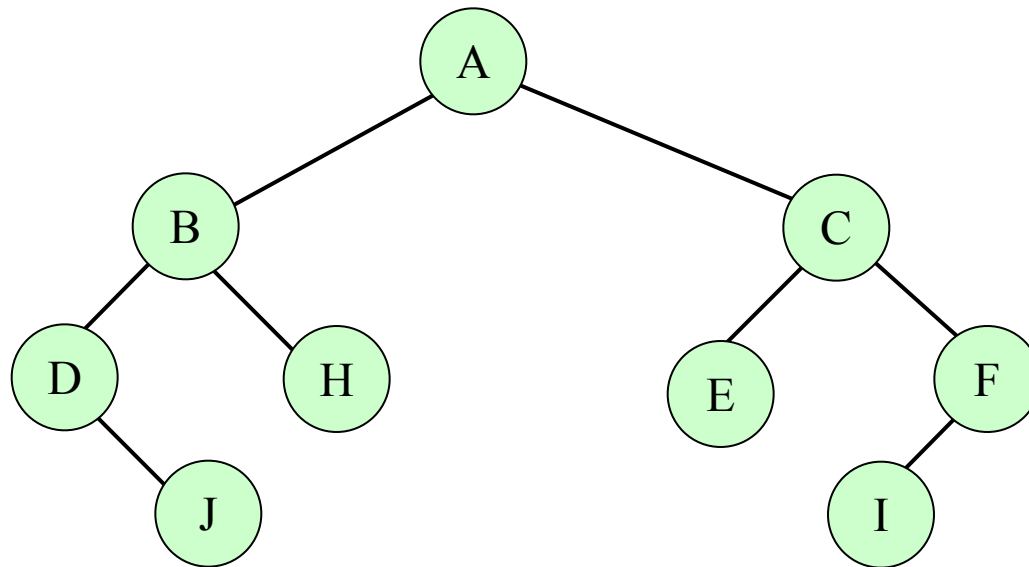
Percurso em Árvore Binária

Ordem Simétrica

percorrer subárvore esquerda da raiz em ordem simétrica

visitar(raiz)

percorrer subárvore direita da raiz em ordem simétrica



Ordem Simétrica: D J B H A E C I F

Percurso em Árvore Binária

Principal

```
se  $pt \neq \lambda$   
  ordemSis( $pt$ )
```

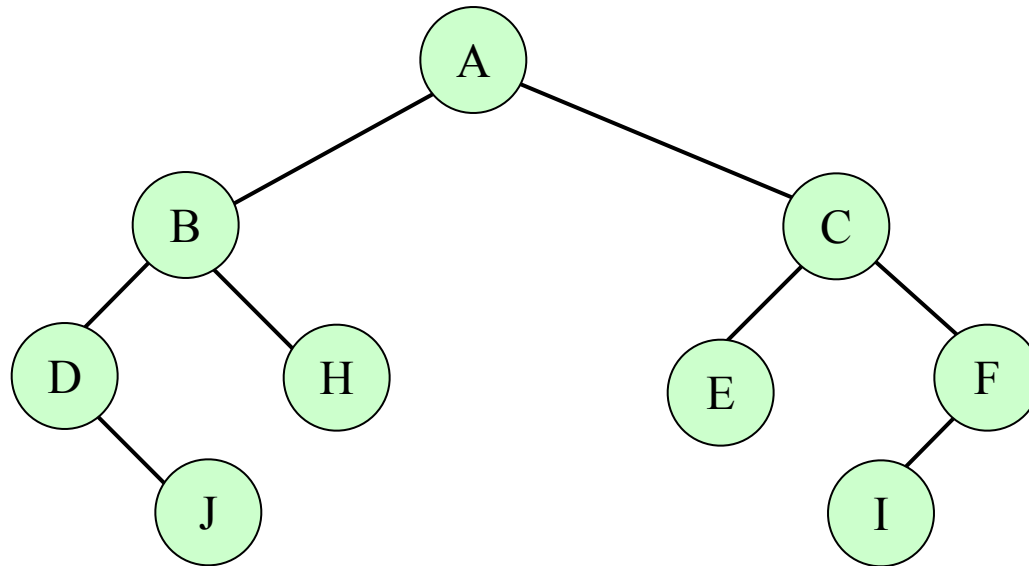
Algoritmo ordemSis(pont_no pt)

```
se  $pt \uparrow .esq \neq \lambda$   
  ordemSis( $pt \uparrow .esq$ )  
visitar( $pt$ )  
se  $pt \uparrow .dir \neq \lambda$   
  ordemSis( $pt \uparrow .dir$ )
```

Percurso em Árvore Binária

Pós ordem

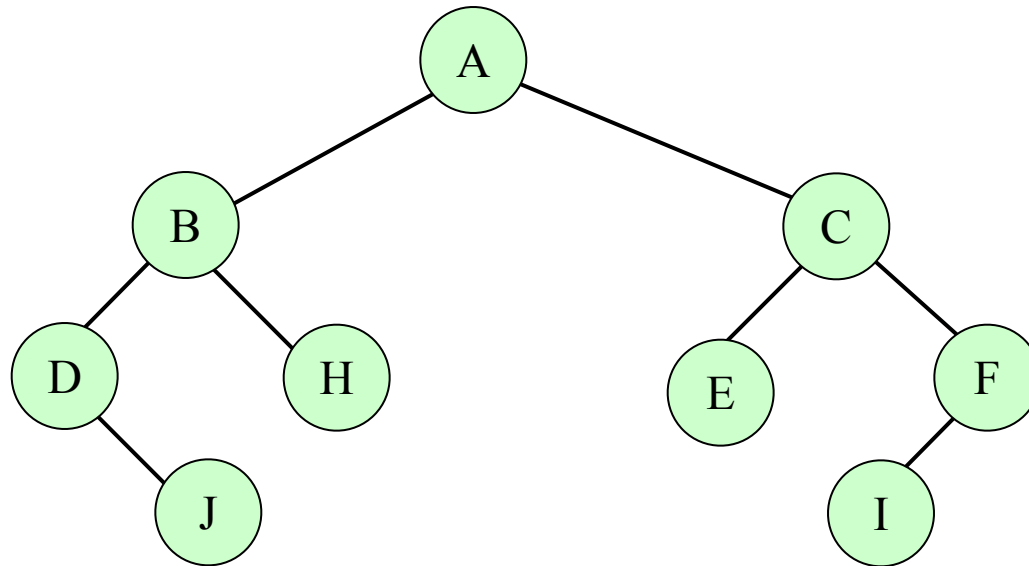
percorrer subárvore esquerda da raiz em pós-ordem
percorrer subárvore direita da raiz em pós-ordem
visitar(raiz)



Percurso em Árvore Binária

Pós ordem

percorrer subárvore esquerda da raiz em pós-ordem
percorrer subárvore direita da raiz em pós-ordem
visitar(raiz)



Pós-Ordem: J D H B E I F C A

Percurso em Árvore Binária

Principal

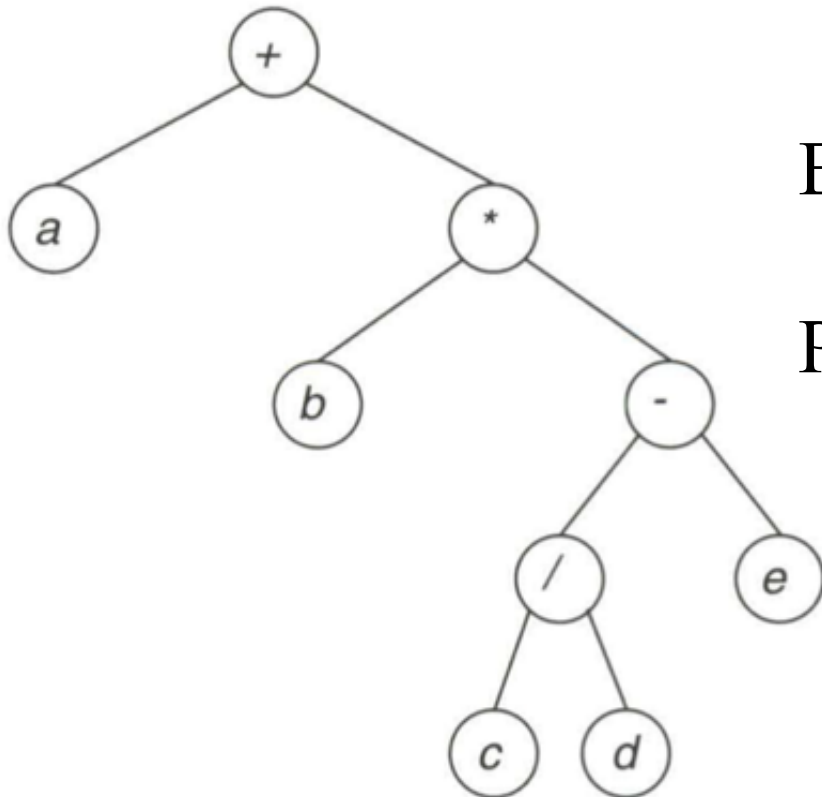
```
se  $pt \neq \lambda$   
  posOrdem( $pt$ )
```

Algoritmo posOrdem(pont_no pt)

```
se  $pt \uparrow .esq \neq \lambda$   
  posOrdem( $pt \uparrow .esq$ )  
se  $pt \uparrow .dir \neq \lambda$   
  posOrdem( $pt \uparrow .dir$ )  
visitar( $pt$ )
```

Expressão aritmética

Qual percurso da árvore abaixo obterá uma expressão polonesa reversa (posfixa)?



Expressão: $(a + (b * ((c / d) - e)))$

Polonesa reversa: **abcd/e-*+**

Exercício Aula

Faça um algoritmo para calcular a altura de todos os nós de uma árvore binária




Exercício Aula

Nó da árvore

Exemplo C:

```
struct nodo {  
    int chave;  
    int altura;   
    struct nodo *esq;  
    struct nodo *dir;  
};  
typedef struct nodo no;  
typedef no *pont_no;
```



Um novo
campo na
nossa estrutura

Exercício Aula

Calcula Altura

Principal

```
se  $pt \neq \lambda$   
  pos_ordem( $pt$ )
```

pos_ordem(pont_no pt)

```
se ( $pt \uparrow$ .esq  $\neq \lambda$ )  
  pos_ordem( $pt \uparrow$ .esq)  
se ( $pt \uparrow$ .dir  $\neq \lambda$ )  
  pos_ordem( $pt \uparrow$ .dir)  
visitar( $pt$ )
```

visita(pont_no pt)

```
sejam alt_e e alt_d inteiros  
se ( $pt \uparrow$ .esq =  $\lambda$ )  
  alt_e  $\leftarrow$  0  
senão alt_e  $\leftarrow pt \uparrow$ .esq $\uparrow$ .altura  
se ( $pt \uparrow$ .dir =  $\lambda$ )  
  alt_d  $\leftarrow$  0  
senão alt_d  $\leftarrow pt \uparrow$ .dir $\uparrow$ .altura  
se ( $alt_e > alt_d$ )  
   $pt \uparrow$ .altura  $\leftarrow pt \uparrow$ .esq $\uparrow$ .altura + 1  
senão  
   $pt \uparrow$ .altura  $\leftarrow pt \uparrow$ .dir $\uparrow$ .altura + 1
```

Árvore Binária



Exercício

Qual a complexidade do algoritmo para calcular a altura de todos os nós da árvore binária?



Percurso em Árvore Binária



Exercícios

Fazer as versões iterativas de:

pre_ordem

ordem_simetrica

pos_ordem



Percurso pré-ordem iterativo

pre_ordem_iterativo (raiz)

Pilha p; bool fim \leftarrow false;

repita

se (raiz $\neq \lambda$)

visita(raiz)

se (raiz \uparrow .dir $\neq \lambda$) então *push*(p, raiz \uparrow .dir)

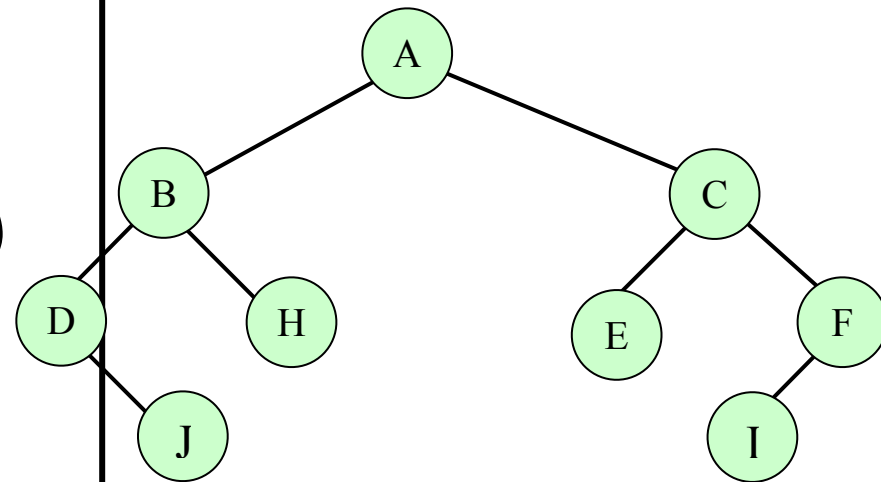
raiz \leftarrow raiz \uparrow .esq

senão

se p é vazia então fim \leftarrow true

senão raiz \leftarrow pop(p)

até fim==true



Pré-ordem: A B D J H C E F I

Percurso de ordem sistemática iterativo

em_ordem_iterativo (raiz)

Pilha p; bool fim \leftarrow false;

repita

 enquanto (raiz $\neq \lambda$)

 empilha (p, raiz); raiz = raiz \uparrow .esq

 se p não vazia então

 raiz \leftarrow pop(p)

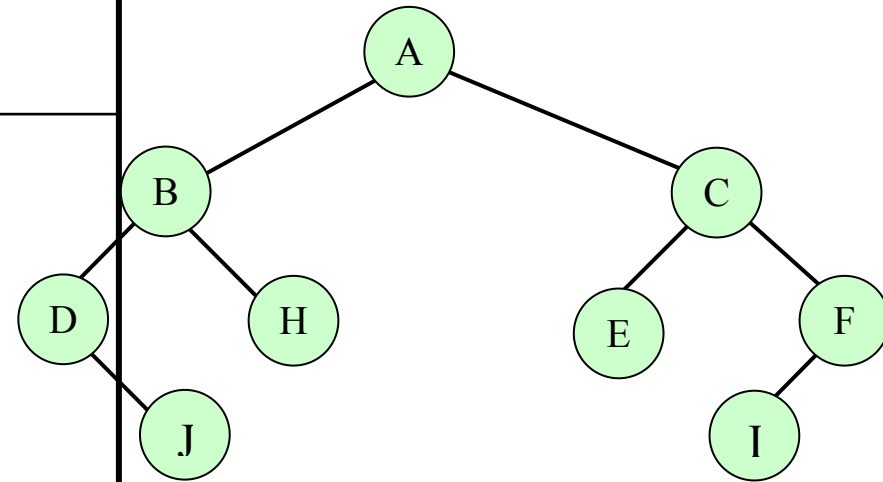
 visita (raiz)

 raiz = raiz \uparrow .dir

 senão

 fim \leftarrow true

até que fim==true



Ordem Simétrica: D J B H A E C I F

Percurso pós-ordem iterativo

pos_ordem_iterativo (raiz)

Pilha p; bool sobre; int m;

repita

enquanto (raiz $\neq \lambda$)

push (p, {raiz, 1}); raiz = raiz \uparrow .esq

sobre \leftarrow true

enquanto (sobre==true && p nao vazia)

{raiz, m} \leftarrow pop(s)

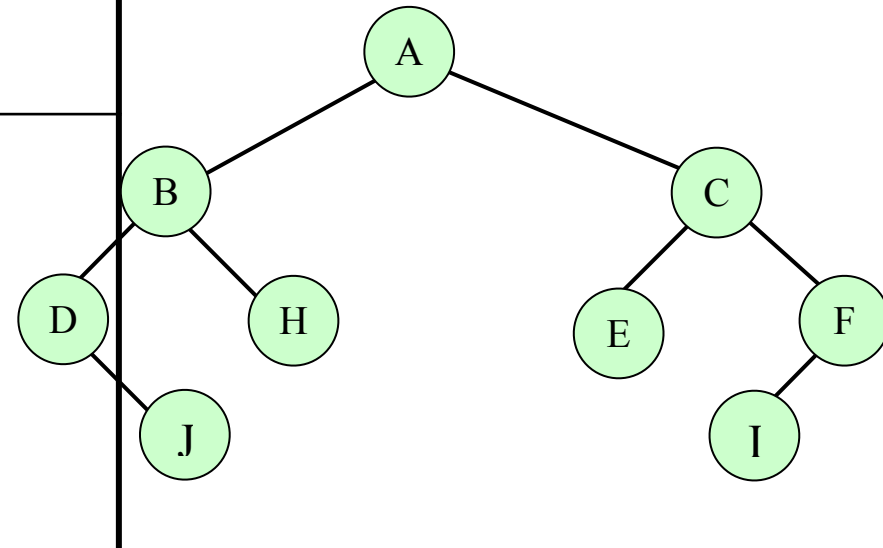
switch(m)

caso 1: push(p, {raiz, 2}); raiz = raiz \uparrow .dir

sobre \leftarrow false;

caso 2: visita(raiz)

até que p esteja vazia



Pós-Ordem: J D H B E I F C A

m = 1 ==> ainda deve empilhar a sub-árvore direita do nó
m = 2 ==> deve visitar o nó

Percurso em Árvore Binária

Percurso em Nível

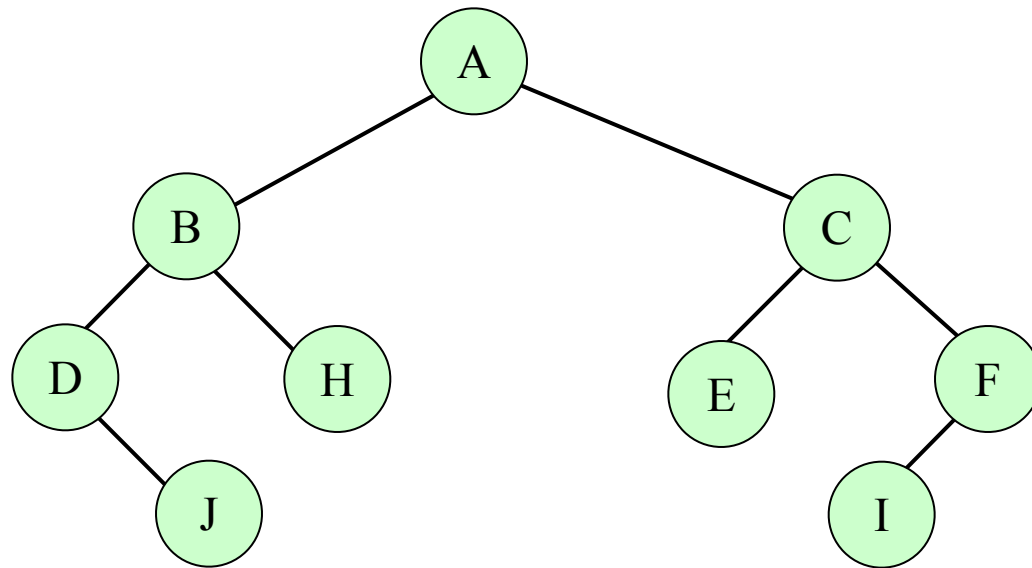
Considerando da esquerda para a direita

Considerando que visitar(raiz) significa imprimir o conteúdo do nó.

Imprime os nós de cada nível da árvore

Percurso em Árvore Binária

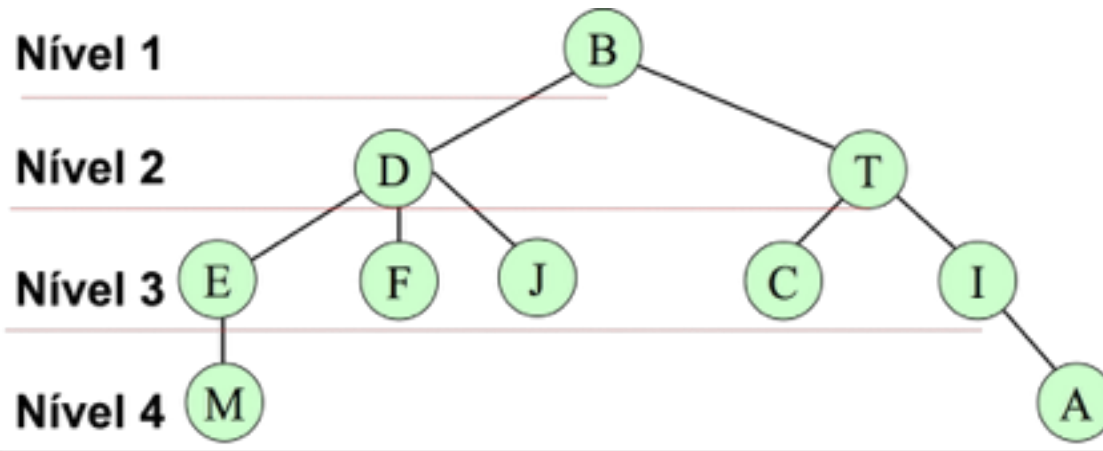
Percurso em Nível



Nível: A B C D H E F J I

Exercício

Faça um algoritmo de percurso em nível de uma árvore TERNÁRIA





FIM
